

ECO quick start 06 - Making the EcoSpace persistent

Table of Contents

Overview	1
Prerequisites and goals	2
XML persistence	3
SQL Server persistence	4
Inspecting the SQL database	6
Polymorphism	7
Other PersistenceMapper types	9
Summary	11
Index	a

1 Overview

An EcoSpace without any persistence is a Transient EcoSpace, meaning that all changes will be lost when the application terminates. If you chose any form of persistence in the ECO wizard your EcoSpace project will have a PersistenceMapperProvider which holds the actual PersistenceMapper component, in this case it will be a PersistenceMapperXml component which saves all data to a local XML file. There are a number of pre-installed PersistenceMappers available from the toolbox including Mimer, MySql, Oracle, and SqlServer.

In addition to the pre-installed PersistenceMapper components there are a number of additional types that ship only as source code. You can read more details about these components in the Other PersistenceMapper types ([see page 9](#)) section of this article.

2 Prerequisites and goals

Prerequisites

To successfully follow this document the user should have read the preceding articles in this series and have a saved project.

Goals

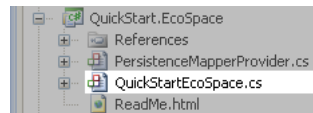
By the end of this document you be able to

- Look briefly at the XML persistence store
- Use an SQL persistence store.

3 XML persistence

The PersistenceMapperXml component is the same component used by the Quick Prototype tool to persist data between testing phases. This persistence mapper will save objects in the EcoSpace to a specific XML formatted file each time the application instructs the EcoSpace to do so. As the entire XML file is read when the EcoSpace is activated and rewritten during saving and this persistence component is really only suitable for use during the development phase, a power failure during the process of updating the file could result in data loss.

1. Open the QuickStart project.
2. In the Solution Explorer window expand the node QuickStart.EcoSpace.
3. Double-click the QuickStartEcoSpace component to bring up its designer.



4. On the EcoSpace you will see that the "PersistenceMapper" property is set to "persistenceMapperSharer1", which is a component on the EcoSpace's designer surface. This is the PersistenceMapper component the EcoSpace will use for all persistence operations such as create, retrieve, update, delete.

Rather than provide the persistence operations itself this PersistenceMapper will delegate them all to the PersistenceMapperProvider. The purpose of this is to allow multiple instances of an EcoSpace to all use a single object instance for their persistence operations. This makes the process of synchronizing changes between multiple EcoSpaces much more simple, especially when the PersistenceMapperProvider is on a remote machine, this technique will be demonstrated in a later article.

5. Select "persistenceMapperSharer1" and look at its "MapperProviderTypeName" property, you will see that it identifies the class name of the PersistenceMapperProvider. In this case the value should be "QuickStart.QuickStartPMP".
6. Now double-click the PersistenceMapperProvider component to bring up its designer.
7. At the moment the project uses XML persistence so you will see a PersistenceMapperXml component on the design surface.

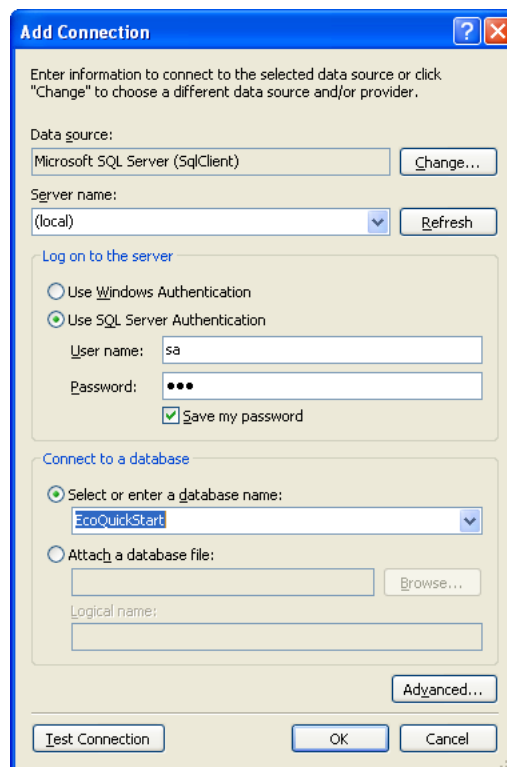
Note that the PersistenceMapperProvider has the two following properties set

1. EcoSpaceTypeName = "QuickStart.QuickStartEcoSpace". The purpose of this is to provide model information to the persistence mapper.
2. PersistenceMapper = "persistenceMapperXml1". This identifies which persistence component to use to service persistence requests. You may have multiple types of PersistenceMapper on the design surface but the PersistenceMapperProvider can only be set to use one of those.

4 SQL Server persistence

The PersistenceMapperSqlServer component allows changes to be persisted to a SQL Server database server. It is possible for ECO to create and maintain the database structure, or to provide object to table mapping information for ECO to update a manually maintained database structure. In this article ECO will be used to create a database with a default structure. This is by far the quickest way of developing applications as it removes the necessity for the developer to maintain the database structure and mapping information.

1. Remove the PersistenceMapperXml1 component from the PersistenceMapperProvider.
2. Create a new SQL Server database named "EcoQuickStart".
3. Drop a PersistenceMapperSqlServer component onto the PersistenceMapperProvider's design surface.
4. Open the dropdown editor for the ConnectionString property and select "New connection"



Note: During design time it is recommended that you use the system administrator account for generating or updating the DB structure. At runtime you should set your connection string to use a non administrator account that has been granted access to the database.

5. Right-click PersistenceMapperSqlServer1 and from the context menu select "SQL Server 2000/2005 setup", this will configure the component so that it is aware of the target database server's reserved words etc.
6. Select the PersistenceMapperProvider in the Object Inspector and ensure that its "PersistenceMapper" property is set to "PersistenceMapperSqlServer1".

The EcoSpace is now configured to use the SQL Server database for persistence. As this is an empty database some tables will be needed for storing the objects.

9. Rebuild your solution to ensure the generated binaries reflect any changes you may have recently made to your model.
10. At the bottom of the PersistenceMapperProvider there is a selection of icons that perform different actions. Click the "Generate Schema" icon.



11. A form will now appear listing tables to delete, tables to recreate, and new tables to be created. Selecting the [New tables] tab will reveal the following tables.

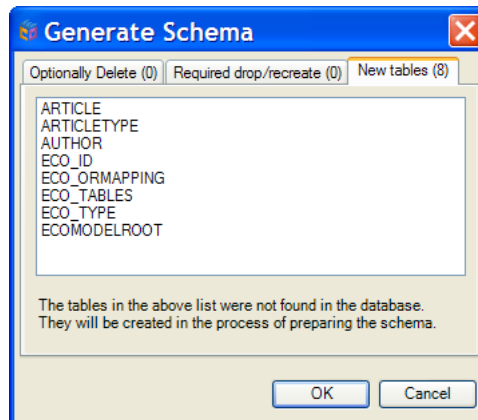
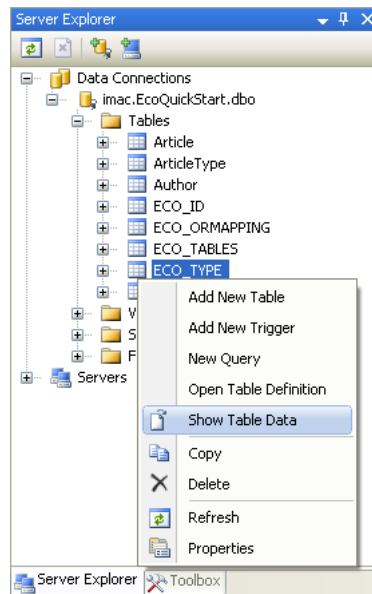


Table name	Description
Article	This table will hold instances of the Article class.
ArticleType	This table will hold instances of the ArticleType class.
Author	This table will hold instances of the Author class.
ECO_ID	This table will contain a single row holding the next Integer value to use as the primary key for new objects. Note: It is possible to use GUIDs for keys, or even a custom mechanism of your own choice.
ECO_OR Mapping	This table will contain a single row holding XML information describing the object to database mapping information.
ECO_TABLES	This table will contain a list of tables created by ECO.
ECO_TYPE	This table will contain a list of modeled class names mapped to integer values, these are used to determine the exact type of an object in a row when inheritance is used.
ECOModelRoot	This table will hold instances of the ECOModelRoot class, this class is ultimately the superclass of all classes within the model. This table holds the ID of the object and an integer value identifying the class type (see ECO_TYPE).

12. Click "OK", the database structure will now be updated.
13. Run the application.
14. Add a new Author.
15. Click the "UpdateDB" button on the form.
16. Restart the application.
17. Note how the objects reappear from the last time the application was executed.

4.1 Inspecting the SQL database

1. If you have not already done so add a connection in your Servers list to the EcoQuickStart database.
2. Expand the database tree nodes until you see the "Tables" node.
3. Right-click the table ECO_TYPE, from the context menu select "Show Table Data".



4. You will now be presented with the following data:

ECO_TYPE	CLASSNAME
0	Author
1	Article
2	ArticleType
3	ECOModelRoot
*	NULL

From this data you can see that the Author class has an ECO_TYPE id of "0" and ArticleType has an ECO_TYPE id of "2".

4. After creating some Authors, ArticleTypes and Articles view all data rows for the ECOModelRoot table:

ECO_ID	ECO_TYPE
1	0
2	0
3	0
4	1
5	1
6	2
*	NULL

5. From this information we can determine that there are 6 object instances in the entire database. By inspecting the ECO_TYPE column we can see that there are three Author instances, two Articles and one ArticleType instance.
6. View all data rows for the Author table to confirm this assumption.

	ECO_ID	ECO_TYPE	ID	EmailAddress	FirstName	LastName	Password	Salutation	AutoPublishArti...
▶	1	0	1	NULL	Kate	Austen	16	Ms	NULL
	2	0	2	NULL	Claire	Littleton	4	Ms	NULL
	3	0	3	NULL	Jack	Shephard	15	Mr	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4.2 Polymorphism

The Latin word "Polymorph" means to have many forms. As ECO allows you to model classes that descend from other classes it is obviously possible via object oriented programming techniques to use polymorphic techniques. For example

- There is a class named "Task" which has a virtual method named "Complete".
- There is a descendant of Task named "BackupDatabaseTask".
- There is another descendant of Task named "RestoreDatabaseTask".

It would be possible to pass an instance of either a BackupDatabaseTask or RestoreDatabaseTask to any method expecting an instance of Task and have the tasks' Complete method executed.

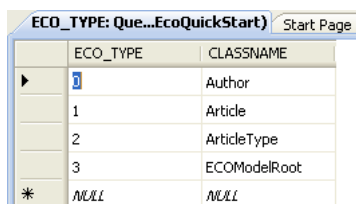
ECO allows the use of such polymorphic techniques when modeling associations between classes, and also when retrieving object instances from the data storage.

- For example an association between a Schedule class and the Task class would allow the application to not only add instances of "Task" to "Schedule.Tasks" but also instances of any class descended from Task. This would allow "Schedule.Tasks" to contain both BackupDatabaseTask and RestoreDatabaseTask instances.
- When retrieving object instances from the data storage the correct type of object is instantiated. Asking for "Task.allInstances" would return a collection of object instances which contains both BackupDatabaseTask and RestoreDatabaseTask objects.
- When iterating though all Tasks in "Schedule.Tasks" you will encounter instances of both BackupDatabaseTask and RestoreDatabaseTask.

When generating a table in the database for a class with no superclass ECO will create an additional column named ECO_TYPE;

	ECO_ID	ECO_TYPE	ID	EmailAddress	FirstName	LastName	Password	Salutation	AutoPublishArti...
▶	1	0	1	NULL	Kate	Austen	16	Ms	NULL
	2	0	2	NULL	Claire	Littleton	4	Ms	NULL
	3	0	3	NULL	Jack	Shephard	15	Mr	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

this identifies the type of the object according to the ECO_TYPE table.



ECO_TYPE	CLASSNAME
	Author
1	Article
2	ArticleType
3	ECOModelRoot
*	NULL

When generating a table in the database for a class that has a superclass the ECO_TYPE column will not be generated. Instead the object instance's data will be split across two tables, for example Task and BackupDatabaseTask, the Task table will contain the ECO_TYPE column plus columns for its own members and the BackupDatabaseTask will contain only columns for its own members. This approach enables the developer to model the application's business classes as if they were part of some kind of persistent memory rather than persisted to a database.

NOTE: It is possible to control the database schema in a number of ways:

- Specifying custom key type, a GUID for example.
- Specifying custom mapping information on a class to identify whether it should have its own table, add its required columns to its parent class's table, or add its required columns to its descendant class's tables (abstract classes only).
- You may provide XML mapping information and map your ECO model to an entirely custom database format.

5 Other PersistenceMapper types

Some ECO PersistenceMapper components ship only as source due to the fact that the database they support may have many releases in a year which makes it difficult for CapableObjects to choose which version to strongly reference before shipping.

The source code for these PersistenceMapper components can be found in the following path on a Windows XP machine.

```
C:\Program Files\CapableObjects\ECO\4.0\source\Persistence
```

At the time of writing this folder contains the following:

- Eco.Persistence.Firebird
- Eco.Persistence.Mimer
- Eco.Persistence.DBX
- Eco.Persistence.MySql
- Eco.Persistence.BlackfishSQL
- Eco.Persistence.NexusDB
- Eco.Persistence.Oracle
- Eco.Persistence.SQLite
- Eco.Persistence.SqlServer
- Eco.Persistence.Sybase

Note: All of these are PersistenceMapper components for RDBMS databases except "Multi". PersistenceMapperMulti enables you to spread your data across multiple databases. These databases may be on different servers or even be different types of database servers. For example you may wish to store User information in a FireBird database on Server01, Order information in a SQL Server database on Server02, and all other information in an Oracle database on Server03.

Using the additional PersistenceMapper components

Using one of the additional PersistenceMapper components is quite simple.

1. open the csproj-file for the persistencemapper you need to build.
2. Build it
3. If there is a Eco.Persistence.XXX.Design project for your persistencemapper, repeat step 1 + 2

Note: Some of the files in these folder are design-time only files and should therefore not be added to the runtime assembly. Typically you should exclude files that contain any of the following words: IDE, Config, or Validate.

Now add the DLL to your toolbox in the usual manner:

1. Right-click the Toolbox

2. Select "Choose Items..."
3. Click the "Browse" button
4. Select the DLL and click "Open"
5. Now scroll through the list until you find the PersistenceMapper and ensure its check box is checked
6. Click the "OK" button.

You can now add the PersistenceMapper to your PersistenceMapperProvider. Whenever a new version of the provider library is made available you can easily modify your project and recompile.

6 Summary

Making an EcoSpace persistent is a very simple process. As a consequence switching to a new database can be achieved in seconds. It is also possible with a small amount of programming to identify which type of database server to connect to at runtime, allowing your application to connect to the database of your customer's choice.

Database structure management

ECO is not only capable of creating your database structure, but also updating the database structure to reflect changes made to your business model. Automatic database structure management saves the developer a considerable amount of time when creating applications.

Supported databases

At the time of writing the following databases are supported by ECO.

- BlackFishSQL.
- DB2 (via Delphi BDP or DBX components).
- InterBase (via Delphi BDP or DBX components).
- FireBird 2.
- Memory (useful for testing).
- MySQL.
- Oracle.
- SQL Server.
- Sybase (via Delphi BDP or DBX components).
- XML file.

More advanced persistence techniques such as mapping to an existing database, reverse engineering an existing database, and using a remote persistence server will be covered in a subsequent document.

Index

I

Inspecting the SQL database 6

O

Other PersistenceMapper types 9

Overview 1

P

Polymorphism 7

Prerequisites and goals 2

S

SQL Server persistence 4

Summary 11

X

XML persistence 3